MySQL für Data Warehousing?

Christian Beilschmidt
areto consulting gmbh
Köln

Schlüsselworte

MySQL, Data Warehousing, Features, Einsetzbarkeit, Vergleich Oracle Datenbank

Einleitung

MySQL ist eines der meistgenutzten Datenbankmanagementsysteme der Welt. Besonders für Webseiten und -anwendungen ist es ein De-facto-Standard. Es zeichnet sich durch seine flexible Erweiterbarkeit aus und Erfuhr zahlreichen Sprach- und Funktionserweiterung in den letzten Jahren.

Oracle selbst bewirbt das Produkt auf der MySQL-Website nicht nur für OLTP-Systeme, sondern auch als Analysesystem für ein Data Warehousing. Dieser Vortrag beschreibt an den Data-Warehouse-Prozessen, wie geeignet MySQL und dessen Features dafür sind. In diesem Sinne wird die Oracle Datenbank als Konkurrent in der Oracle-eigenen Produktpalette als Referenz herangezogen.

Wieso gibt es die Idee, MySQL im Data Warehousing einzusetzen?

Oracle bietet seit Jahren Datenbanksysteme (zuletzt 11g und neuerdings 12c) an, die sich über Jahre in vielen Data-Warehouse-Projekten bewährt haben. Deshalb stellt sich die Frage zur Motivation, wieso MySQL als Alternative erwogen werden sollte. Der Hauptgrund liegt hierbei in den Lizenzkosten, die bei MySQL im Vergleich zur Oracle Datenbank vergleichsweise günstig ausfallen.

Da MySQL open-source Software ist, ist die Verwendung prinzipiell kostenfrei. Wer den Support in Anspruch nehmen möchte, muss die Enterprise-Version lizensieren. Der Standard-Edition fehlen Features wie Partitionierung, die wichtig sind und später genauer betrachtet werden. Auch zusätzliche Enterprise-Tools zur einfacheren Verwaltung des Gesamtsystems machen die Lizensierung erforderlich und stellen sinnvolle Erweiterungen dar. Je nach Größe des Projekts ist die MySQL *Cluster Carrier Grade Edition* notwendig, um eine verteilte Datenbank zu verwalten.

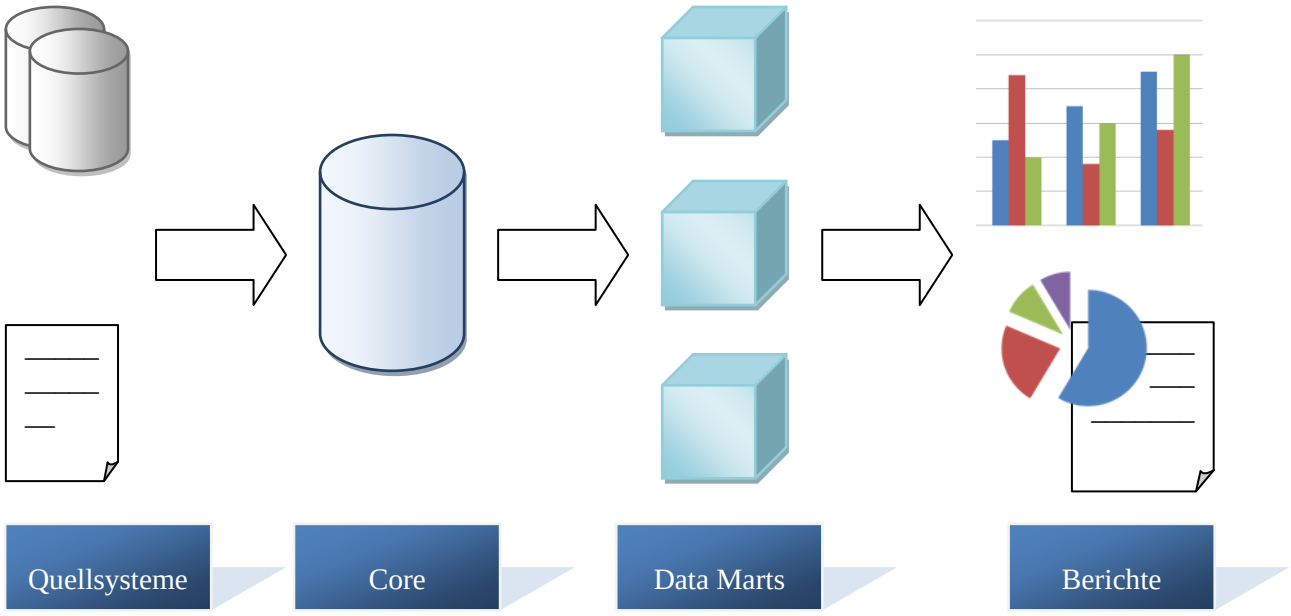

Abb. 1: Data-Warehouse-Prozesse

Da die Lizensierung insbesondere bei Oracle sehr komplex ist, können keine genauen Zahlen gegenübergestellt werden. Generell kann jedoch festgestellt werden, dass sich alle Features bei Oracle in zusätzlichen Kosten niederschlagen, während MySQL entweder kostenfrei ist oder in der Enterpriseversion direkt sämtliche Features vorhanden sind. Diese Preisunterschiede rechtfertigen es nachzuprüfen, ob MySQL eine Alternative für das Data Warehousing darstellt.

Data-Warehouse-Prozesse

Um zu überprüfen, ob MySQL ein geeignetes Produkt im Data-Warehouse-Umfeld ist, müssen Anforderungen herausgearbeitet und die Umsetzbarkeit bewertet werden. Hierfür ist es hilfreich, die Prozesse im Data Warehouse zu betrachten und dort einzelne Aspekte aufzugreifen. Abbildung 1 zeigt Prozesse, die in einem Projekt vorhanden sind. Hierbei sind die Aufgaben chronologisch rückwärts zu betrachten. Zu Beginn gibt es den Wunsch, bestimmte Berichte beispielsweise über ein Unternehmen und dessen Prozesse zu sehen. Dies können wiederkehrende Berichte sein, die z.B. täglich generiert werden oder Ad-hoc-Berichte, die Antworten auf spontane Fragen liefern sollen. Hierfür ist es notwendig, dass die Daten in geeigneter Weise dafür verfügbar und gespeichert sind. Diese Daten befinden sich in den Data Marts, die über effiziente interne Strukturen verfügen, um diese Berichte generieren zu können. Diese Daten befinden sich wiederum im Core des Data Warehouse, in dem sämtliche analytisch verwertbaren Unternehmensdaten in der kleinstmöglichen Granularität vorliegen. Data Marts werden unter anderem nach Kriterien wie Abteilungen oder Sicherheitsaspekten aus den Daten aus dem Core erstellt. Die Daten des Cores stammen letztendlich aus Quellsystemen, die Transaktionsdaten aus den Unternehmensprozessen bereitstellen. Es sind die Daten, die analysiert werden sollen. Viele Unternehmen verwenden mehrere Transaktionssysteme, die konsolidiert werden müssen. Dieser Prozess wird ETL-Prozess genannt, wobei **e**xtract, **t**ransfer und **l**oad die Vorgänge beschreibt, wie Daten aus den Quellsystemen in das Data Warehouse gelangen.

Damit MySQL für ein Data Warehouse in Frage kommt, muss es möglich sein alle Schritte damit zu realisieren. Das heißt konkret, dass bestimmte Features vorhanden sein müssen und andere Anforderungen vielleicht nur über Umwege zu erfüllen sind. Als Vergleich kann die Oracle Datenbank herangezogen werden, die alle Features beinhaltet und somit als Status Quo dient.

Name	Default?	Transaktional?	Besonderheiten
MyISAM	Ja	Nein	Volltextsuche, schnell Lesen
InnoDB	Ja	Ja	Referentielle Integrität
Memory	Ja	Nein	In-Memory-Speicher
BerkleyDB	Nein	Ja	Key-Value Store
Falcon	Nein	Ja	Starkes Caching von Inserts
Archive	Ja	Nein	Speicheroptimiert, ohne Indexe
CSV	Ja	Nein	Kommaseparierte Textdateien
Federated	Ja	Ja	Externe Tabellen
NDB Cluster	Ja	Ja	Verteilte Tabellen
InfiniDB	Nein	Ja	Column-Store, Analyse-Engine
TokuDB	Nein	Ja	Schreiboptimierter B-Baum
XtraDB	Nein	Ja	InnoDB fork
Infobright	Nein	Ja	Column-Store, automatische Indexerstellung *Knowledge Grid*

Tabelle 1: MySQL Storage-Engines

MySQL Engines

Im Gegensatz zur Oracle Datenbank verwendet MySQL nicht eine einzelne Speichertechnologie. Vielmehr ist MySQL seit jeher mit austauschbaren Speicherengines versehen, die jeweils pro Tabelle ausgewählt werden können. Dies erlaubt die Auswahl der passenden Engine für eine bestimmte Aufgabe.

Tabelle 1 listet einige Engines auf, wobei viele standardmäßig in MySQL verfügbar sind. Andere Engines sind Produkte anderer Hersteller, welche in MySQL integrierbar sind. Interessant sind hierbei insbesondere die Produkte InfiniDB und Infobright, die eine spaltenorientierte Speicherung anbieten. Dies ist für Analysedatenbanken von Vorteil, da bei Aggregationen lediglich die verwendeten Spalten von der Festplatte geladen werden müssen. Eine zeilenweise Speicherung erfordert das Laden der jeweils gesamten Zeile und ist eher für das transaktionelle Umfeld geeignet, wo einzelne Einträge bearbeitet und gelesen werden. Da Oracle auch in der Oracle Datenbank verglichen mit anderen Marktführenden Datenbanksystemen weiterhin keine Spaltenspeicherung anbietet, liegt hier der Vorteil bei MySQL aufgrund der Flexibilität.

ETL-Prozess

Ladeprozesse erfordern das Konsolidieren mehrerer Quellsysteme. Hierfür ist es praktisch, wenn Konnektoren zu anderen System bereits verfügbar sind. MySQL bietet zwei Engines an, die dabei hilfreich sind. Die Erste ist die Federated Engine, die das Einbinden externer Tabellen in ein Schema erlaubt. Das andere Datenbanksystem muss hierbei allerdings auch MySQL sein. CSV-Dateien können ebenfalls mit der entsprechenden Engine eingebunden werden. Um aus anderen Quellsystemen Daten zu laden, müssen die Verbindungen manuell erstellt werden. ETL-Tools sind hierfür sinnvoll und auch für MySQL verfügbar. Beispiele sind open-source Lösungen wie CloverETL und Talend OpenStudio wie auch der Oracle Warehouse Builder und der Oracle Data Integrator. Letztere erlauben mittels JDBC ebenfalls MySQL als verwendetes Datenbanksystem, müssen allerdings von Oracle lizensiert werden.

Um die Integrität und Konsistenz der Daten zu gewährleisten, kann es hilfreich sein, Semantik in den Schemadefinitionen zu spezifizieren. Dies ist beispielsweise mit Constraints oder referentieller Integrität möglich, die in MySQL beispielsweise mit InnoDB verwendet werden können. Dies erleichtert eine Überprüfung, wobei fehlerhafte Datensätze bereinigt werden müssen. Ferner bietet InnoDB Transaktionen nach ACID, die bei komplexeren ETL-Prozessen sinnvoll sein können, in diesem Schritt allerdings seltener eingesetzt werden. Viele Prozesse setzen eher auf Kontrollfunktionen innerhalb der Prozess-Skripte. MySQL bietet seit Version 5.7 wie auch die Oracle Datenbank eine eigene Skriptsprache, die es erlaubt Statements zu persistieren. Der Sprachumfang ist noch nicht so groß wie bei der Oracle Datenbank, sodass oftmals weiterhin auf eine andere Sprache ausgewichen werden muss. In der Praxis gibt es für jede moderne Programmiersprache einen Konnektor zu MySQL, meistens über ODBC.

MySQL ist im Gegensatz zur Oracle Datenbank kein objektrelationales Datenbanksystem. D.h. es können keine User-Defined-Types erstellt werden, die den Impedence-Mismatch zu objektorientieren Systemen vereinfachen.

Kernschema

Im Core wird in diesem Fall die multidimensionale Modellierung betrachtet. Hierbei werden die Daten entweder im Snowflakeschema oder denormalisiert im Starschema gespeichert. MySQL bietet hier wie die Oracle Datenbank ausreichend Möglichkeiten, die Beziehungen zwischen Dimensionen und Faktentabellen zu beschreiben. Das Grundschema ist aus diesem Grund nicht unterschiedlich.

Wichtige Komponenten sind Indexierung und Partitionierung, die einen Zugriff auf die Daten beschleunigen und die Organisation vereinfachen sollen. MySQL bietet wie die Oracle Datenbank eine Möglichkeit, Tabellen horizontal nach Hash-Sets, Listen oder Bereichen zu partitionieren. Vertikale Partitionierung wird bei beiden Systemen nicht unterstützt, könnte aber durch manuelles Aufsplitten der Tabellen erreicht werden. Dieses Vorgehen würde Sinn geben, wenn Joins zweier Faktentabellensplits ausgeschlossen sind. Andernfalls wird die Performance deutlich sinken und eine Column-Store-Engine ist vorzuziehen, die dieses Problem auf eine andere Weise löst. Diese gibt es für MySQL. Subpartitionierung wird genau wie eine Partitionierung von Indexen unterstützt, womit bei MySQL etwas weniger Features im Detail bietet als die Oracle Datenbank. Ein wichtiges Merkmal wie der Austausch von Partitionen ist allerdings möglich. Ebenfalls greifen Abfragen nur auf die Partitionen einer Tabelle zu, die sich mit den Angaben im Where-Teil decken.

An Indextypen bietet MySQL B-Bäume, allerdings keinen Bitmapindex, der in anderen Systemen oftmals als Join-Index zwischen der Fakten- und einer Dimensionstabelle verwendet wird. Function-based Indexe sind nur eingeschränkt möglich, da lediglich auf Prefixe oder ähnliches zugegriffen werden kann.

Das Backup des Data Warehouse ist wichtig und sollte so unkompliziert wie möglich sein, ist jedoch kein Hauptaugenmerk des Vortrags. Die Oracle Datenbank stellt mit dem Recovery Manager (RMAN) ein ausgereiftes Tool bereit und auch MySQL bietet in der Enterpriseversion ebenfalls ein Enterprise Backup an. Dieses Tool erlaubt Hot-Backups, bei der das Datenbanksystem nicht heruntergefahren werden muss, sowie inkrementelle Backup. Dies und eine eingebaute Kompression reduzieren das aufkommende Datenvolumen des Backups. Mittels mitgesicherter Logdateien ist eine exakte Point-In-Time-Recovery möglich.

Data Marts

Data Marts stellen prinzipiell eine Teilmenge des Cores da und sind stärker auf einen effizienten Zugriff optimiert. Deshalb ist es positiv, dass MySQL eine Kompression anbietet, um schnelle Abfragen aufgrund einer besseren I/O- und CPU-Auslastung zu gewährleisten. Um die Leistung der Kompression zu verbessern, muss die Art der Daten und die Kompressionsstärke untersucht werden. Hierfür können verschiedene Blockgrößen getestet werden. Kleinere Blöcke können durch weniger Leseoperationen mehr Daten liefern, solange der Prozessor diese Daten ausreichend schnell dekomprimieren kann.

MySQL stellt mit der Memory-Engine einen einfachen In-Memory-Speicher zur Verfügung, um bestimmte Daten dauerhaft im Hauptspeicher vorzuhalten. Die fehlende Transaktionalität ist aufgrund des Read-only-Zugriffs nicht nachteilhaft. Indexe sind weiterhin möglich, zusätzlich gibt es sogar einen Hash-Index.

Materialized Views stellen in der Oracle Datenbank eine gute Möglichkeit dar, einfach eine Redundanz zur Abfrageoptimierung in das System einzubauen, die per Befehl auf den aktuellen Stand gebracht werden kann. In MySQL muss Redundanz manuell eingefügt werden. Dies ist mit überschaubarem Aufwand möglich. Es wird eine weitere Tabelle erstellt, die semantisch gesehen die View ersetzt. Zur ersten Beladung der Tabelle kann die Query einfach ausgeführt werden. Bei Änderungen ist es performancetechnisch unerwünscht, alle Daten neu zu berechnen. Je nach Abfrage (z.B. Aggregationstyp) können neue Daten von alten Daten, die nicht verändert werden müssen, getrennt werden. Hier ist es dann Möglich die Query um eine zeitliche Bereichsabfrage zu erweitern und ausschließlich neue Daten einzufügen. Kompliziertere Änderungen erfordern deduktive Mechanismen, womit prinzipiell manuell der Mechanismus von der Oracle Datenbank nachgebildet wird. Aufgrund der großen Community existieren einige Standardverfahren, die adaptiert werden können. Der Query Re-Write ist so allerdings nicht möglich, dieses Feature ist auch nicht einfach

nachrüstbar. Somit muss bei den Views im Auge behalten werden, dass Reporting Tools diese nutzen. Das macht Ad-hoc-Reporting kompliziert, da Anwender diese Optimierung selbst nachvollziehen müssten.

Query Performance

Der MySQL Query Optimizer hat im Gegensatz zum Cost-based Optimizer von der Oracle Datenbank eher Probleme, komplexe Queries für die Abfragen zu cachen. Aus diesem Grund wird bei älteren MySQL Versionen empfohlen, komplexe Subselects aus Abfragen in einfachere Abfragen mittels temporären Tabellen herunterzubrechen und dem Optimizer die Chance zu geben, eine gute Cachingstrategie zu verwenden. Je nach Datenmenge in den Dimensionen kann es sinnvoll sein, das Data-Warehouse-Schema eher in Richtung Snowflake zu entwickeln, da teilweise Leistungseinbrüche durch große Relationen im Join entstehen. TPC-H Benchmarks innerhalb der Community zeigen Verbesserungen in den neuen Versionen, die diese Probleme bei vielen Queries beheben.

Mit Shard-Query ist derzeit eine weitere Query-Engine in aktiver Entwicklung, die über Umwege der Federated Engine parallele Query-Verarbeitung unterstützt. Hierbei werden Aggregationen über partitionierte Tabellenteile aufgeteilt und parallel verarbeitet. MySQL an sich verwendet niemals mehr als eine CPU bzw. einen Prozess oder Thread pro Abfrage. Dieses Bottleneck kann mit Shard-Query umgangen werden.

Analytische Funktionen wie Window-Functions sind Features, die in MySQL fehlen um komplexere Analyseanfragen zu formulieren, die effizient verarbeitet werden können. Es ist möglich, diese Queries einerseits über Subselects und Joins zu formulieren, andererseits die Funktion *GROUP_CONCAT()* zu verwenden. Erstere Methode ist deutlich weniger Performant als analytische Formulierungen und letztere Methode ist ein Workaround, der nicht immer möglich ist. Listing 1 zeigt ein Beispiel, in dem die LAG-Funktion mittels Subquery ersetzt werden kann. Hierbei wird jeweils das aktuelle und vorherige Gehalt eines Mitarbeiters ausgegeben. Der Subquery beinhaltet wie die Window-Funktionen ein *ORDER-BY* und gibt durch das *LIMIT* ebenfalls nur den vorherigen Wert zurück.

```
NAME          MONTH         SALARY
-----------------------------------
Mike          1             3500
Mike          2             3700
Lilly         1             1200
Lilly         2             1100

SELECT month, name, salary,
       LAG(salary) OVER (PARTITION BY name ORDER BY month)
  FROM emp_sal;

SELECT month, name, salary,
       (select salary from emp_sal b where a.name=b.name
        and a.month > b.month order by month limit 1)
  FROM emp_sal a;
```

Listing 1: Umschreiben eines analytischen Queries

Reporting Tools

Für das Reporting bietet MySQL viele Möglichkeiten. So gibt es Open-Source-Projekte wie Petanho oder JasperReports von JasperSoft, die eine grafische Aufbereitung der Analysedaten ermöglichen. Außerdem sind gängige kommerzielle Produkte wie Crystal Reports oder MicroStrategy möglich. Auch weitere Produkte wie IBM Cognos und SAP BusinessObjects können sich mit MySQL verbinden, da oftmals ODBC als Konnektor verwendet werden kann.

Fazit

Abschließend kann festgestellt werden, dass MySQL alle essentiellen Features bietet, um eine Data Warehouse zu realisieren. Im Vergleich zur Oracle Datenbank gibt es wenige Features, die nicht realisiert werden können. Dies führt allerdings teilweise dazu, dass mehrere externe Komponenten oder Umwege zusammen verwendet werden müssen, wo die Oracle Datenbank eine Komplettlösung bietet. Die große Community von MySQL verspricht rasche Weiterentwicklungen auch auf dem OLAP-Gebiet. Da MySQL selbst Data Warehousing bewirbt, wird wahrscheinlich ein gewisser Fokus auf diesen Anwendungsfällen auch in Zukunft liegen. Große Data-Warehouse-Projekte könnten einige Feature vermissen, die insbesondere die Leistung komplexerer Anfragen verbessert. In MySQL ist hingegen vieles über Umwege zu erreichen, sodass ein kreatives Entwicklerteam erforderlich ist, da es oftmals noch keine Standardverfahren gibt. Hier ist ein gewissenhaftes Vorgehen notwendig, um diese entstehende Komplexität zu verwalten. Ferner ist es wichtig, dass die benötigten Features des Systems klar herausgearbeitet werden und die entsprechenden Funktionen von MySQL bekannt sind. Interessant ist MySQL insbesondere für kleinere oder mittelgroße Projekte, die evtl. deutlich günstigere Lizensierungskosten in etwas höhere Entwicklungskosten umverteilen können und möchten.

Kontaktadresse:
Christian Beilschmidt
areto consulting gmbh
Julius-Bau-Straße 2
D-51063 Köln

Telefon:	+49 (0) 221 66 95 75-0
Fax:	+49 (0) 221 66 95 75-99
E-Mail	christian.beilschmidt@areto-consulting.de
Internet:	www.areto-consulting.de